

BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL


```
BBBBBBBBB      AAAAAA      SSSSSSSS      PPPPPPPP      UU      UU      RRRRRRRR      IIIIII      000000      BBBBBBBB
BBBBBBBBB      AAAAAA      SSSSSSSS      PPPPPPPP      UU      UU      RRRRRRRR      IIIIII      000000      BBBBBBBB
BB      BB      AA      AA      SS      PP      PP      RR      RR      II      00      00      BB      BB
BB      BB      AA      AA      SS      PP      PP      RR      RR      II      00      00      BB      BB
BB      BB      AA      AA      SS      PP      PP      RR      RR      II      00      00      BB      BB
BBBBBBBBB      AA      AA      SSSSSS      PPPPPPPP      UU      UU      RRRRRRRR      II      00      00      BBBBBBBB
BBBBBBBBB      AA      AA      SSSSSS      PPPPPPPP      UU      UU      RRRRRRRR      II      00      00      BBBBBBBB
BB      BB      AAAAAAAAAA      SS      PP      PP      RR      RR      II      00      00      BB      BB
BB      BB      AAAAAAAAAA      SS      PP      PP      RR      RR      II      00      00      BB      BB
BB      BB      AA      AA      SS      PP      PP      RR      RR      II      00      00      BB      BB
BB      BB      AA      AA      SS      PP      PP      RR      RR      II      00      00      BB      BB
BBBBBBBBB      AA      AA      SSSSSSSS      PP      UU      UU      RR      RR      IIIIII      000000      BBBBBBBB
BBBBBBBBB      AA      AA      SSSSSSSS      PP      UU      UU      RR      RR      IIIIII      000000      BBBBBBBB

LL      IIIIII      SSSSSSSS
LL      IIIIII      SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLLL      IIIIII      SSSSSSSS
```



```

1 0001 0 MODULE BAS$$PUR_IO_BUF (
2 0002 0 IDENT = '1-007'
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1 ++
31 0031 1 FACILITY:
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1 This module contains routines which will check the LUN indicated by R11
36 0036 1 and print the contents of the associated I/O buffer if there is valid
37 0037 1 data in the buffer. These routines are intended to be called before
38 0038 1 a file is closed (explicitly or implicitly at end of program) or if
39 0039 1 an error occurs during an output element transmit.
40 0040 1
41 0041 1 ENVIRONMENT: User mode - AST reentrant
42 0042 1
43 0043 1 AUTHOR: Donald G. Petersen, CREATION DATE: 22-Jan-79
44 0044 1
45 0045 1 MODIFIED BY:
46 0046 1
47 0047 1 DGP, : VERSION 1-01
48 0048 1 1-001 - original
49 0049 1 1-002 - Use 32-bit addresses for externals. JBS 27-JAN-1979
50 0050 1 1-003 - Change entry point names so we have two: one for CLOSE
51 0051 1 and one for error handling. JBS for DGP 07-MAR-1979
52 0052 1 1-004 - Make PUR_IO_ERR purge the terminal on unit zero. DGP 07-Mar-79
53 0053 1 1-005 - Remove references to BAS$ routines. JBS 10-MAY-1979
54 0054 1 1-006 - Change from an OTS routine to a BAS routine, since FORTRAN
55 0055 1 does not need to purge I/O buffers. JBS 20-AUG-1979
56 0056 1 1-007 - Don't purge virtual arrays; it is quite complex, and CLOSE
57 0057 1 will do it. JBS 30-AUG-1979

```

```

! Purge I/O buffer for a file.
! File: BASPURI0B.B32

```


BAS\$\$PUR_IO_BUF
1-007

I 16
16-Sep-1984 00:58:52
14-Sep-1984 11:56:32

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASPURI0B.B32;1

Page 2
(1)

:	58	0058	1	!--
:	59	0059	1	
:	60	0060	1	!<BLF/PAGE>


```
62 0061 1 !
63 0062 1 ! SWITCHES:
64 0063 1 !
65 0064 1 !
66 0065 1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
67 0066 1 !
68 0067 1 !
69 0068 1 ! LINKAGES:
70 0069 1 !
71 0070 1 !
72 0071 1 REQUIRE 'RTLIN:OTSLNK'; ! define all linkages
73 0500 1 !
74 0501 1 !
75 0502 1 ! TABLE OF CONTENTS:
76 0503 1 !
77 0504 1 !
78 0505 1 FORWARD ROUTINE
79 0506 1     BAS$$PUR_IO_ERR : NOVALUE, ! Purge outstanding output buffer
80 0507 1 ! following an error
81 0508 1     BAS$$PUR_IO_CLO : CALL_CCB NOVALUE; ! Purge outstanding output buffer
82 0509 1 !
83 0510 1 ! during a close
84 0511 1 !
85 0512 1 ! INCLUDE FILES:
86 0513 1 !
87 0514 1 !
88 0515 1 REQUIRE 'RTLML:OTSLUB'; ! I/O statement block
89 0655 1 !
90 0656 1 REQUIRE 'RTLIN:RTLPSECT'; ! Define DECLARE_PSECTS macro
91 0751 1 !
92 0752 1 REQUIRE 'RTLIN:BASIOERR'; ! Define I/O error symbols
93 0805 1 !
94 0806 1 LIBRARY 'RTLSTARLE'; ! STARLET library
95 0807 1 !
96 0808 1 !
97 0809 1 ! MACROS:
98 0810 1 !
99 0811 1 !     NONE
100 0812 1 !
101 0813 1 ! EQUATED SYMBOLS:
102 0814 1 !
103 0815 1 !
104 0816 1 LITERAL
105 0817 1     K_CR = %X'0D'; ! ASCII CR
106 0818 1     K_NULL = %X'00'; ! ASCII NUL
107 0819 1     K_LF = %X'0B'; ! ASCII LF
108 0820 1 !
109 0821 1 !
110 0822 1 ! PSECT declarations
111 0823 1 !
112 0824 1 DECLARE_PSECTS (BAS); ! Put this in BAS psect
113 0825 1 !
114 0826 1 ! OWN STORAGE:
115 0827 1 !
116 0828 1 !     NONE
117 0829 1 !
118 0830 1 ! EXTERNAL REFERENCES:
```


BAS\$\$PUR_IO_BUF
1-007

K 16
16-Sep-1984 00:58:52
14-Sep-1984 11:56:32

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASPURIOB.B32;1

Page 4
(2)

```
: 119      0831 1 !  
: 120      0832 1  
: 121      0833 1 EXTERNAL ROUTINE  
: 122      0834 1 BAS$$CB_PUSH : JSB CB_PUSH,  
: 123      0835 1 BAS$$CB_POP : JSB CB_POP NOVALUE,  
: 124      0836 1 BAS$$NEXT_LUN : NOVALUE,  
: 125      0837 1 BAS$$STOP_IO : NOVALUE;  
: 126      0838 1
```

```
: Load register CCB  
: Done with register CCB  
: Get next logical unit number  
: Signal fatal BASIC I/O error
```



```
128 0839 1 GLOBAL ROUTINE BASS$PUR_IO_CLO          ! Purge the contents of an I/O buffer
129 0840 1      : CALL_CCB NOVALUE =
130 0841 1
131 0842 1
132 0843 1 ++
133 0844 1 FUNCTIONAL DESCRIPTION:
134 0845 1      This routine will PUT the contents of the buffer associated with the LUN
135 0846 1      passed to it if the buffer has valid data in it. This routine is expected
136 0847 1      to aid in closing a file either implicitly or explicitly if the last lan-
137 0848 1      guage to access the file has the Basic semantics where an output record
138 0849 1      may be continued across several output statements. Therefore, an output
139 0850 1      buffer may exist which has valid data in it which should be printed before
140 0851 1      a program finishes executing.
141 0852 1
142 0853 1 FORMAL PARAMETERS:
143 0854 1
144 0855 1     NONE
145 0856 1
146 0857 1 IMPLICIT INPUTS:
147 0858 1
148 0859 1     LUB$V_OUTBUF_DR      Flag to indicate the output buffer has valid
149 0860 1                        data in it
150 0861 1     LUB$W_LUN          LUN number
151 0862 1
152 0863 1 IMPLICIT OUTPUTS:
153 0864 1
154 0865 1     LUB$V_OUTBUF_DR      Flag to indicate the output buffer has valid
155 0866 1     LUB$B_BAS_VFC2       'post' carriage control for PRN file format
156 0867 1                        data in it
157 0868 1
158 0869 1 ROUTINE VALUE:
159 0870 1
160 0871 1     NONE
161 0872 1
162 0873 1 SIDE EFFECTS:
163 0874 1
164 0875 1     NONE
165 0876 1
166 0877 1 --
167 0878 1
168 0879 2 BEGIN
169 0880 2
170 0881 2 EXTERNAL REGISTER
171 0882 2     CCB : REF BLOCK [, BYTE];
172 0883 2
173 0884 3 IF (.CCB [LUB$V_OUTBUF_DR] AND (.CCB [LUB$B_ORGAN] NEQ LUB$K_ORG_VIRTU))
174 0885 2 THEN
175 0886 3 BEGIN
176 0887 3 ++
177 0888 3 Write out the buffer. This involves a call to RMS.
178 0889 3 --
179 0890 3     CCB [RAB$W_RSZ] = .CCB [LUB$A_BUF_PTR] - .CCB [LUB$A_RBUF_ADR];
180 0891 3     CCB [RAB$L_RBF] = .CCB [LUB$A_RBUF_ADR];
181 0892 3     CCB [LUB$V_OUTBUF_DR] = 0;
182 0893 3
183 0894 4 IF ( NOT $PUT (RAB = .CCB))
184 0895 3 THEN
```



```
!End of BAS$$PUR_IO_CLO
```

```
.TITLE  BAS$$PUR_IO_BUF
.IDENT  \1-007\

.EXTRN  BAS$$CB_PUSH, BAS$$CB_POP
.EXTRN  BAS$$NEXT_LUN, BAS$$STOP_IO
.EXTRN  SYSS$PUT, SYSS$WAIT

.PSECT  _BAS$CODE, NOWRT, SHR, PIC, 2
```

					0004	0000	.ENTRY	BASS\$PUR_IO-CLO, Save R2	: 0839
		52	00000000G	00	9E	00002	MOVAB	SYSS\$PUT,-R2	:
5C	FE	AB		03	E1	00009	BBC	#3, -2(CCB) 4\$	: 0884
		05	C4	AB	91	0000E	CMPB	-60(CCB), #5	:
				56	13	00012	BEQL	4\$	:
22	AB	B0	AB	EC	AB	A3	SUBW3	-20(CCB), -80(CCB), 34(CCB)	: 0890
		28	AB	EC	AB	D0	MOVL	-20(CCB), 40(CCB)	: 0891
		FE	AB		08	8A	BICB2	#8, -2(CCB)	: 0892
					5B	DD	PUSHL	CCB	: 0894
		62			01	FB	CALLS	#1, SYSS\$PUT	:
		28			50	E8	BLBS	R0, 3\$	:
	000182DA	8F	08	AB	D1	0002C	CMPL	8(CCB), #99034	: 0898
				10	12	00034	BNEQ	2\$	:
				5B	DD	00036	PUSHL	CCB	: 0900
	00000000G	00		01	FB	00038	CALLS	#1, SYSS\$WAIT	:
				5B	DD	0003F	PUSHL	CCB	: 0901
		62		01	FB	00041	CALLS	#1, SYSS\$PUT	:
				E6	11	00044	BRB	1\$	: 0896
		0A	08	AB	E8	00046	BLBS	8(CCB), 3\$	: 0904
		7E		01	CE	0004A	MNEGL	#1, -(SP)	:
	00000000G	00		01	FB	0004D	CALLS	#1, BASS\$\$STOP_IO	:
	DA	AB		0B	B0	00054	MOVW	#11, -38(CCB)-	: 0908

BASS\$PUR_10_BUF
1-007

B 1
16-Sep-1984 00:58:52
14-Sep-1984 11:56:32

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASPURI0B.B32;1

Page 7
(3)

B0	AB	EC	AB	D0	00058	MOVL	-20(CCB), -80(CCB)	:	0913
	50	D2	AB	3C	0005D	MOVZWL	-46(CCB), R0	:	0914
B4	AB	EC	BB40	9E	00061	MOVAB	@-20(CCB)[R0], -76(CCB)	:	
		C8	AB	D4	00067	CLRL	-56(CCB)	:	0915
				04	0006A	4\$:	RET	:	0919

; Routine Size: 107 bytes, Routine Base: _BASS\$CODE + 0000

; 209 0920 1


```
211 0921 1 GLOBAL ROUTINE BAS$$PUR_IO_ERR          ! Purge terminal I/O
212 0922 1 : NOVALUE =
213 0923 1
214 0924 1 ++
215 0925 1 FUNCTIONAL DESCRIPTION:
216 0926 1
217 0927 1 This routine will PUT the contents of any terminal buffer.
218 0928 1 It is used just before printing an error message to be sure that
219 0929 1 the message appears after any output produced before the error
220 0930 1 condition.
221 0931 1
222 0932 1 FORMAL PARAMETERS:
223 0933 1
224 0934 1 NONE
225 0935 1
226 0936 1 IMPLICIT INPUTS:
227 0937 1
228 0938 1 LUB$V_OUTBUF_DR          Flag to indicate the output buffer has valid
229 0939 1                          data in it
230 0940 1
231 0941 1 IMPLICIT OUTPUTS:
232 0942 1
233 0943 1 LUB$V_OUTBUF_DR          Flag to indicate the output buffer has valid
234 0944 1 LUB$B_BAS_VFC2          'post' carriage control for PRN file format
235 0945 1                          data in it
236 0946 1
237 0947 1 ROUTINE VALUE:
238 0948 1
239 0949 1 NONE
240 0950 1
241 0951 1 SIDE EFFECTS:
242 0952 1
243 0953 1 NONE
244 0954 1
245 0955 1 --
246 0956 1
247 0957 2 BEGIN
248 0958 2
249 0959 2 GLOBAL REGISTER
250 0960 2 CCB = K_CCB_REG : REF BLOCK [, BYTE];
251 0961 2
252 0962 2 LOCAL
253 0963 2 FLAG,
254 0964 2 LUN;
255 0965 2
256 0966 2 ++
257 0967 2 Scan through all logical units, purging the ones OPEN to a terminal.
258 0968 2 --
259 0969 2 FLAG = 0;
260 0970 2
261 0971 2 DO
262 0972 2 BEGIN
263 0973 2 ++
264 0974 2 Get the next logical unit number.
265 0975 2 --
266 0976 2 BAS$$NEXT_LUN (FLAG, LUN);
267 0977 2
```



```

: 268      0978 4      IF (.FLAG NEQ 0)
: 269      0979 3      THEN
: 270      0980 4      BEGIN
: 271      0981 4      !+
: 272      0982 4      ! LUN is the next logical unit number. If the file it represents is
: 273      0983 4      ! open to a terminal, purge it.
: 274      0984 4      !-
: 275      0985 4      BAS$$CB_PUSH (.LUN, LUB$K_ILUN_MIN);
: 276      0986 4
: 277      0987 5      IF (.CCB [LUB$V_OPENED] AND .CCB [LUB$V_UNIT_0] AND .CCB [LUB$V_FORCIBLE])
: 278      0988 4      THEN
: 279      0989 4      BAS$$PUR_IO_CLO ();
: 280      0990 4
: 281      0991 4      BAS$$CB_POP ();
: 282      0992 3      END;
: 283      0993 3
: 284      0994 3      END
: 285      0995 2      UNTIL (.FLAG EQL 0);
: 286      0996 2
: 287      0997 2      RETURN;
: 288      0998 1      END;

```

!End of BAS\$\$PUR_IO_ERR

5E	08	0804 00000	.ENTRY	BAS\$\$PUR_IO_ERR, Save R2,R11	: 0921
	04	AE C2 00002	SUBL2	#8, SP	: 0969
	08	AE D4 00005	CLRL	FLAG	: 0976
	08	5E DD 00008 1\$:	PUSHL	SP	
00000000G 00	04	AE 9F 0000A	PUSHAB	FLAG	
	04	02 FB 0000D	CALLS	#2, BAS\$\$NEXT_LUN	: 0978
	25	AE D5 00014	TSTL	FLAG	
50	08	13 00017	BEQL	3\$	: 0985
52	08	CE 00019	MNEGL	#8, R0	
	6E	D0 0001C	MOVL	LUN, R2	
00000000G 00	00	16 0001F	JSB	BAS\$\$CB_PUSH	: 0987
OF	FC	AB E9 00025	BLBC	-4(CCB), 2\$	
	FE	AB 95 00029	TSTB	-2(CCB)	
	0A	18 0002C	BGEQ	2\$	
05	FE	06 E1 0002E	BBC	#6, -2(CCB), 2\$	
FF5D	AB	00 FB 00033	CALLS	#0, BAS\$\$PUR_IO_CLO	: 0989
	CF	00 16 00038 2\$:	JSB	BAS\$\$CB_POP	: 0991
	04	AE D5 0003E 3\$:	TSTL	FLAG	: 0995
	C5	12 00041	BNEQ	1\$	
	04	00043	RET		: 0998

; Routine Size: 68 bytes, Routine Base: _BAS\$CODE + 006B

```

: 289      0999 1
: 290      1000 1 END
: 291      1001 1
: 292      1002 0 ELUDOM

```

!End of module - BAS\$\$PUR_IO_BUF

PSECT SUMMARY

```

:
:      Name                Bytes                Attributes
:
:  _BAS$CODE              175  NOVEC,NOWRT, RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
:

```

Library Statistics

```

:
:      File                ----- Symbols ----- Pages Processing
:                        Total   Loaded   Percent   Mapped   Time
:
:  _$255$DUA28:[SYSLIB]STARLET.L32;1    9776         9         0       581     00:01.2
:

```

COMMAND QUALIFIERS

```

:
:      BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:BASPURI0B/OBJ=OBJ$:BASPURI0B MSRC$:BASPURI0B/UPDATE=(ENH$:BASPURI0B
:      )
:
:  Size:          175 code + 0 data bytes
:  Run Time:      00:09.1
:  Elapsed Time:  00:22.4
:  Lines/CPU Min: 6577
:  Lexemes/CPU-Min: 35960
:  Memory Used:  103 pages
:  Compilation Complete

```


0029 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

BASOPEN
LIS

BASPOWJ
LIS

BASPOS
LIS

BASPOWJ
LIS

BASOPENDE
LIS

BASPOWGG
LIS

BASPOWH
LIS

BASPOWRJ
LIS

BASPOWII
LIS

BASPURJOB
LIS

BASPOWDO
LIS

BASOPENZE
LIS

BASPOWR
LIS

BASPOWJ
LIS

BASPOWR
LIS

BASPOWH
LIS

BASPOWR
LIS

0030 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

